

# LINKEDLIST

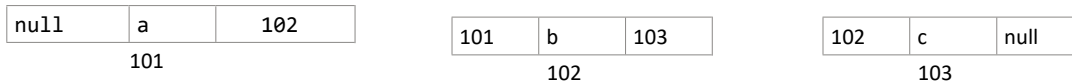
01 March 2025 13:23

## What is LinkedList?

- LinkedList is a **doubly linked list implementation** in Java.

Previous node address	element	Next node address
-----------------------	---------	-------------------

```
list.add("a");  
list.add("b");  
list.add("c");
```



- It is part of **Java Collection Framework**.
- It **implements both List and Deque interfaces**, so it supports **list operations and queue operations** both.
- It is **part of java.util package**.
- It **allows duplicates** and **preserves insertion order**.
- It allows **null values**.

## Why do we need LinkedList?



### Problem with ArrayList

- In ArrayList, inserting or removing elements **in the middle** is **slow** because shifting is required.



### Solution - LinkedList

- In LinkedList, **no shifting required**.
- It uses **nodes**, where each node stores:
  - **Data**
  - **Link to next node**
  - **Link to previous node**

## Key Features of LinkedList



Doubly linked list structure



No shifting when adding/removing in middle



Maintains **insertion order**



Allows **duplicates**



Allows **null**



Supports **both list (index-based) and deque (queue methods)**



# Common Methods of LinkedList

Method	Description
<code>add(E e)</code>	Add element at end
<code>add(int index, E e)</code>	Add at specific index
<code>get(int index)</code>	Get element at index
<code>set(int index, E e)</code>	Update element at index
<code>remove(int index)</code>	Remove element at index
<code>remove(Object o)</code>	Remove specific element
<code>size()</code>	Get number of elements
<code>contains(Object o)</code>	Check if element exists
<code>isEmpty()</code>	Check if list is empty
<code>clear()</code>	Remove all elements
<code>toArray()</code>	Convert to array
<code>addFirst(E e)</code>	Add element at start
<code>addLast(E e)</code>	Add element at end
<code>removeFirst()</code>	Remove first element
<code>removeLast()</code>	Remove last element
<code>getFirst()</code>	Get first element

## Creating LinkedList (Syntax)

```
import java.util.LinkedList;
LinkedList<String> list = new LinkedList<>();
```

## Internal Working of LinkedList

- Internally uses **doubly linked nodes**.
- Each node has:
  - data (actual element)
  - next (link to next node)
  - prev (link to previous node)
- No shifting during insert/remove — only links are changed.

## Advantages of LinkedList

- ✓ Fast insertion/removal from **beginning/middle/end**
- ✓ No need to resize
- ✓ Can work like **List** and **Queue/Deque** both
- ✓ Doubly linked for two-way traversal

## Disadvantages of LinkedList

- ✗ More memory (each node has 2 extra pointers — next, prev)
- ✗ Slower random access (no index array like ArrayList)

## ◆ When to Use LinkedList?

Use Case	Recommended?
✓ Insert/remove frequently (especially in middle)	✓ Yes
✓ Need both List & Queue features	✓ Yes
✗ Fast random access needed	✗ No (Use ArrayList)
✗ Memory is a concern	✗ No

## ◆ Quick Difference: ArrayList vs LinkedList

Feature	ArrayList	LinkedList
Internal Structure	Array	Doubly Linked List
Memory	Compact	More (extra pointers)
Access Speed	Fast (direct index)	Slow (traverse from head)
Insert/Remove (Middle)	Slow (shifting needed)	Fast (just change links)
Insert/Remove (Start/End)	Slow	Fast
Part of	List interface	List + Deque interface



## ◆ Real World Example for Explanation

Scenario	Best Choice
✓ Passenger Queue	LinkedList (FIFO nature)
✓ Students Roll Numbers List	ArrayList (Index access needed)
✓ Undo History in Text Editor	LinkedList (Sequential backward/forward traversal)

### CODE :

```
package collection_framework;
import java.util.LinkedList;

public class LinkedListDemo {
    public static void main(String[] args) {
        LinkedList<String> list = new LinkedList<>();
    }
}
```

```

list.add("Apple");
list.add("Banana");
list.add("Mango");

System.out.println("Original List: " + list);

list.addFirst("Orange");
list.addLast("Grapes");

System.out.println("After addFirst and addLast: " + list);

list.removeFirst();
list.removeLast();

System.out.println("After removeFirst and removeLast: " + list);

System.out.println("First element: " + list.getFirst());
System.out.println("Last element: " + list.getLast());

System.out.println("Contains Mango? " + list.contains("Mango"));
System.out.println("Index of Banana: " + list.indexOf("Banana"));

list.set(1, "Kiwi");
System.out.println("After set(1, Kiwi): " + list);

list.remove("Apple");
System.out.println("After removing Apple: " + list);

System.out.println("Size: " + list.size());

list.clear();
System.out.println("After clear, isEmpty? " + list.isEmpty());
}
}

```